Predictive Modeling for Tire Sales Data at All American Tires LLC

Nancy Lizeth Jaramillo

Advisor: Dr. Piryatinska

San Francisco State University

December 10, 2025

Contents

1 Introduction

All American Tires LLC is an independently owned tire shop located in Riverside, California. While they are a leading local company, they recognized the need to leverage their historical sales data to optimize operations and strategically plan for future stability. Their main services include the sale of new and used tires, in addition to comprehensive services like installation, balancing, rotation, flat repairs, alignments, and general mechanics.

My internship began with two primary objectives aimed at modernizing the company's data infrastructure and leveraging its sales history. The first objective was to address an inefficient, paper-based workflow by designing and implementing a new online invoice tracking system. This system allows employees to seamlessly create new invoices, save them to an internal database, and instantly email them to the customer.

With the new system in place and a clean dataset collected, the second objective was to conduct a comprehensive Exploratory Data Analysis. This analysis focused on answering crucial business questions. Some questions being considered were: What days of the week and months generate the highest revenue? Which car brands are the most frequent customers? Is it economically beneficial to operate on Sundays or holidays?

Finally, the third objective involved performing predictive modeling on the dataset. This advanced analysis used various classification and resampling techniques to identify key factors influencing sales and to build models that could project future business stability, providing a data-driven foundation for long-term strategic decision-making.

2 Data Description and Collection

During the summer, I worked as a Data Scientist for All American Tires in Riverside, CA. My role could be explained in two parts: online inventory analysis and predictive modeling. However, before I could do any analysis, I needed to create my data set. I was given a year's worth of paper invoices to turn into a digital archive. This tedious task took about a month to complete because I had to enter every entry by hand. Once everything was inputted into a spreadsheet, I had to clean the data. Some had missing values and others had too much information. An example being, someone wrote down their address but left out their zip code. Once this was complete, I was able to create an online invoice maker for the company to use. I made sure to include some buttons that automatically store the invoice in a database as well as clear the page for easy access.

Then I was able to begin modifying a copy of the original data set to better suit my modeling and analysis. One of the big changes I made was combining zip codes into categories based on which county they belong to. I also added some columns based on the given data. An example of this was creating a binary variable for whether or not the customer purchased tires rather than having a variable that tells you how many tires they purchased. Another important variable I added to the data set was the average temperature of the day of purchase. I added this because I was interested to see whether or not the this had a significant impact on tires purchased. I added this value to the temperature column by hand by using the national weather service and searching through their archived sheets. In total, there were about 1850 customers and 10 variables per customer.

3 Exploratory Data Analysis

The first variable used was temperature. This variable describes the local temperature in degrees Fahrenheit during the day the customer came into the shop. The temperature ranges from fifty-three to one hundred thirteen with a mean of 74 degrees. Below is a visual of the most common temperatures throughout the year as well as how many times during the year they occurred.

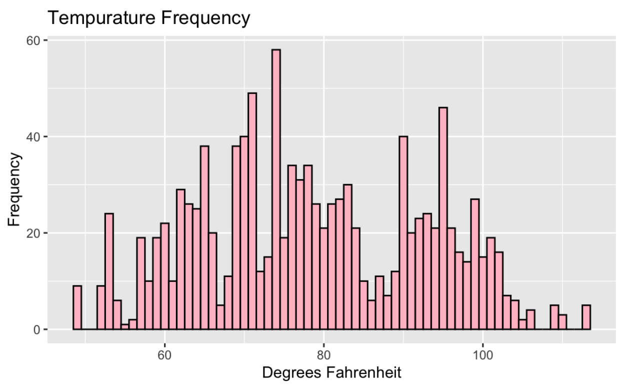


Figure 1: How common the average temperature was during 2024 sales days. Temperature in degrees Fahrenheit of the day customers came in.

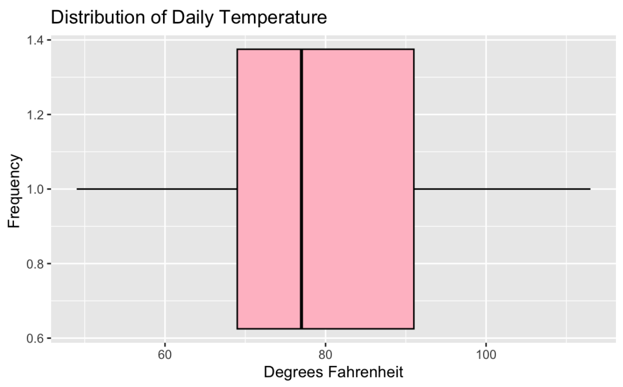


Figure 2: Daily temperature on days customers made purchases

Another important variable to consider was the type of car each customer brought in to be serviced. There were thirty-six different models brought in throughout the year. The most popular car type was Toyota. The least popular car type was a Jetta. Below is a distribution that shows the amount of each model that was brought in throughout the year.

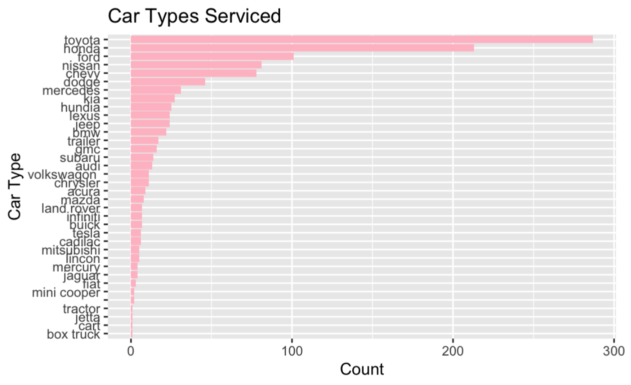


Figure 3: Amount of each car type brought in 2024

Total is also an important variable for this analysis. It tells us how much money each customer spent. The mean value found in total was $438.80 while the median was $300. The minimum spent by one customer was $0 while the maximum spent by one customer was $5,420. The zero dollars was caused by someone just coming in to get their tires filled with air, which the shop does not charge for. The maximum spent by one customer was caused by a smaller company purchasing some inventory from All American Tires. Below is a distribution of the most common totals along with how many times they occurred throughout the year.

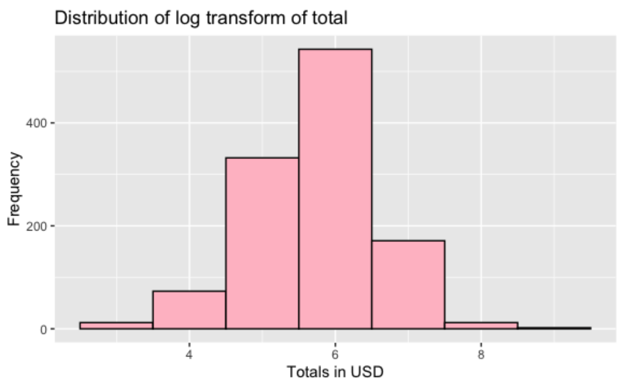


Figure 4: A log base 10 transformation of total amount of money spent by each customer in hundreds USD

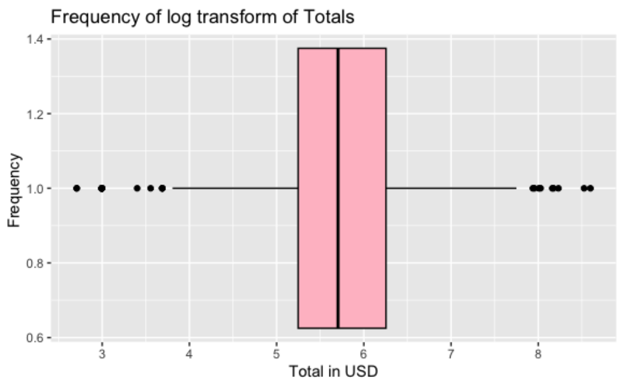


Figure 5: A log base 10 transformation of total amount of money spent by each customer in hundreds USD

Mechanical services are also a large part of All American Tires. While it did not normally lead to tire sales, I decided to include it because it contributes largely to the total distribution of money customers paid. I broke down the data into a binary that states whether or not the customer had mechanical services done on the day they brought in their car. Figure 4 shows their distributions.

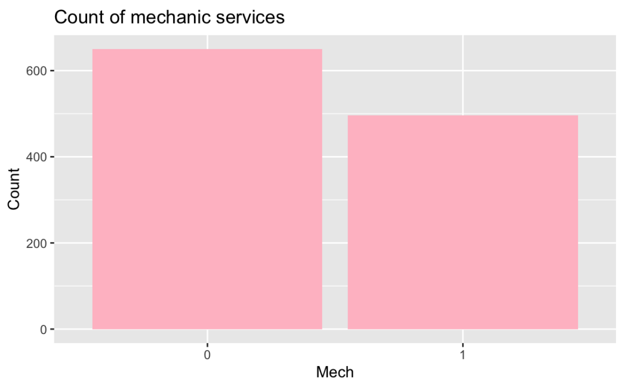


Figure 6: Amount of customers who needed mechanical services in 2024

Similarly, I added a variable that accounts for both alignments and rotations. I combined these into one variable because of the similarities in services and count.

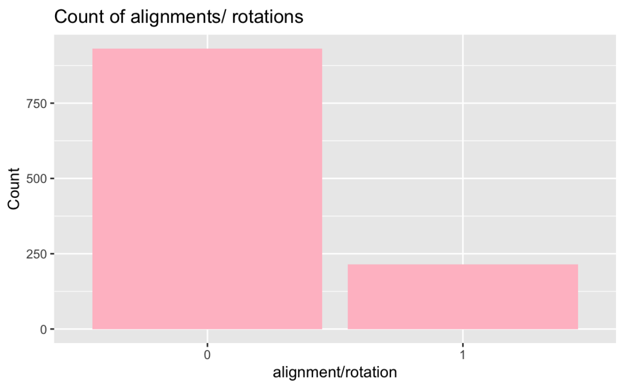


Figure 7: How many customers in 2024 had an alignment/rotation done at time of service

To have a better sense of what contributed to the total amount spent and when tires were purchased more often, I put the purchases in categories of day of the week, month, and whether or not it was a holiday. Figure 6 shows how many customers came in on each day of the week throughout the year 2024. Figure 7 shows how many customers came in each month during 2024. This was intended to help analyze which months produced the highest revenue. There were some holidays that the shop decided to stay open in attempt to increase sales. Figure 8 shows how many customers came in on a holiday.

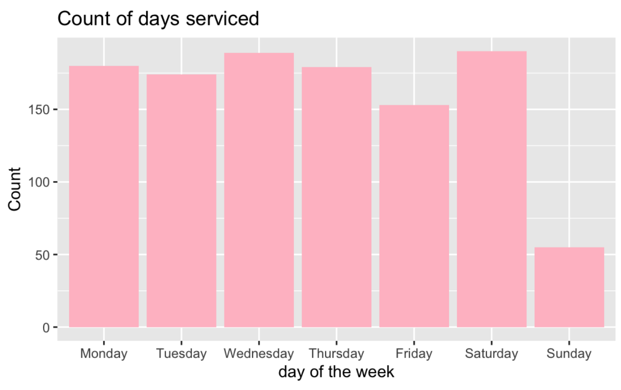


Figure 8: Customer count by day of the week

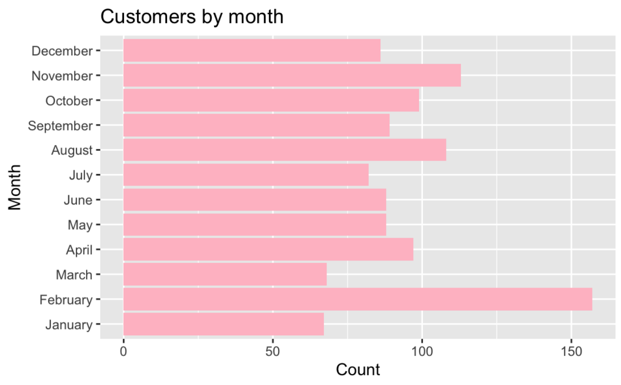


Figure 9: Customer count by month

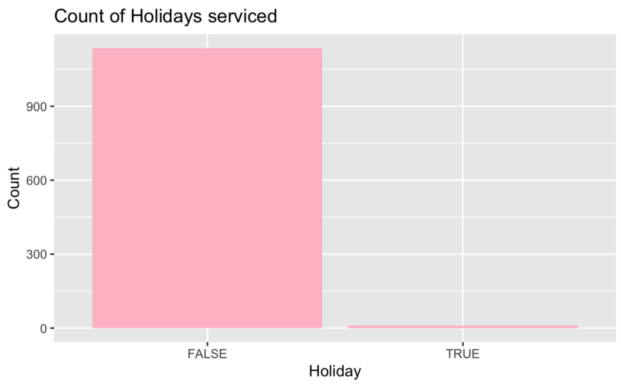


Figure 10: Customers by holiday

Tires purchased shows how many customers purchased tires in 2024. Rather than show how many tires each customer purchased on a scale of one to five, I decided to create a binary variable that simply states whether or not the customer purchased tires. Figure 9 shows the comparison of customers who purchased tires and those who did not.

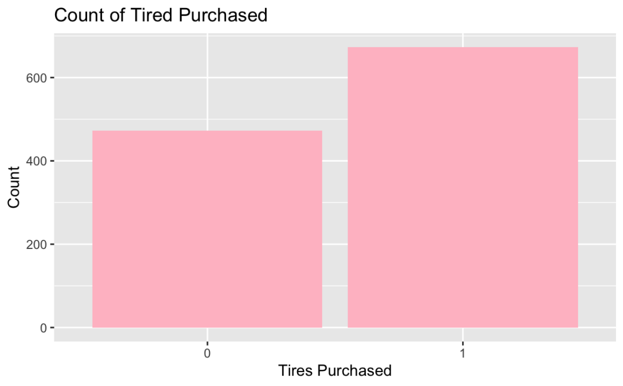


Figure 11: Customers by whether or not they purchased tires.

One of the most important factors to the company besides revenue was where the majority of their customers were coming from. This was important for many reasons including advertising, reputation, and reach. There were too many area codes to create a clean visualization, so I decided to group them by county. Figure 10 shows the how many customers came from which counties. The most common counties are Riverside and Los Angeles. All American Tires is located in Riverside, so this makes sense. There are some outliers that can be explained by recent moves or wrong addresses listed.

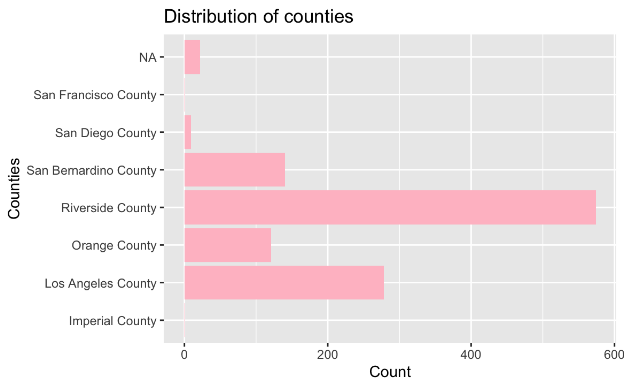
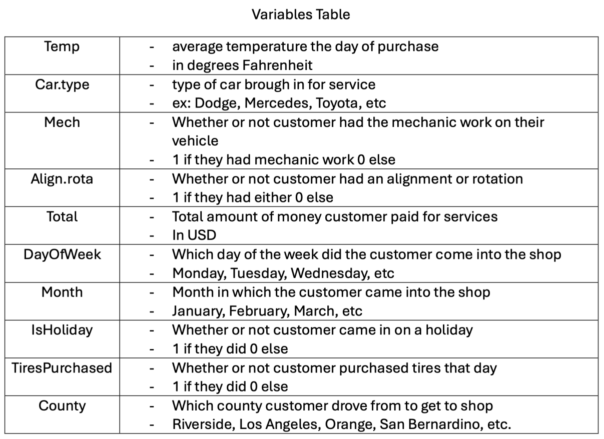


Figure 12: Customer by county

The following table better describes what each variable is and how it was measured.


Variables Table

Temp	- average temperature the day of purchase - in degrees Fahrenheit
Car.type	- type of car brough in for service - ex: Dodge, Mercedes, Toyota, etc
Mech	- Whether or not customer had the mechanic work on their vehicle - 1 if they had mechanic work 0 else
Align.rota	- Whether or not customer had an alignment or rotation - 1 if they had either 0 else
Total	- Total amount of money customer paid for services - In USD
DayOfWeek	- Which day of the week did the customer come into the shop - Monday, Tuesday, Wednesday, etc
Month	- Month in which the customer came into the shop - January, February, March, etc
IsHoliday	- Whether or not customer came in on a holiday - 1 if they did 0 else
TiresPurchased	- Whether or not customer purchased tires that day - 1 if they did 0 else
County	- Which county customer drove from to get to shop - Riverside, Los Angeles, Orange, San Bernardino, etc.

Figure 13: Description of what each variable is and how it was measured in the data set

4 Methodology

4.1 Classification

Classification is a task in machine learning. It can be supervised or unsupervised, but here we will only apply supervised learning classification. Its goal is to predict a categorical label or class for each input in the given data set (James et al. (2021)) A trivial example of classification that most adults are familiar with is spam mail. When an email is received, the platform (Apple, Google, Yahoo, etc.) classifies it as spam or not based on certain factors. Classification is a supervised learning task because it used labeled data to train the model to make accurate predictions. In our spam mail example, emails labeled spam and not spam were initially input to train the program. During this training phase, the program found common traits in each that it would later use as deciding factors to classify future emails.

Once you have a classification model, it is important to evaluate its performance. Evaluating the model's performance is more than just looking at accuracy. It is encouraged to create a table that summarizes the performance by comparing the labels it predicted with the actual labels. Let us consider a two by two matrix. The matrix will show the number of correctly labeled positives, correctly labeled negatives, incorrectly labeled positives, and incorrectly labeled negatives. When the model incorrectly labels an entry as positive, we call this type 1 error. When the model incorrectly labels an entry as negative, we call this type 2 error. Another factor to consider is the trade off between variance and bias. The variance is the amount by which the model would change if we estimated it using a different training set. Ideally, the model would not change too much based on the training data set to produce accurate classifications. High variance can lead to high changes in classification due to minor changes in the training data set. Similarly, bias refers to the error that is caused by using a simple model for a complicated real-life problem. It is important to remember that when models are more flexible, the variance will increase and the bias will decrease. A flexible machine learning model being one that can adapt to complex problems.

discuss: cross validation or split data into training / test set

4.1.1 LDA/QDA

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction and classification technique. It finds a lower-dimensional subspace that maximizes the separation between class means while simultaneously minimizing the variation within each class. LDA models the distribution of the predictors X separately for each value of Y. It then uses Bayes' theorem to flip them into estimates. The goal of Bayes' classification rule is to classify an observation x in the class k that maximizes the posterior probability $P(Y = k|X = x)$.

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

where π_k is the prior probability and $f_k(x)$ is the class-conditional density. LDA is the simpler of the two discriminant analysis. It assumes that the class-conditional densities $f_k(x)$ are Multivariate Gaussian, and that all classes share the same covariance matrix with the resulting classification boundary being a linear function of x. The discriminant function $\delta_k(x)$ is

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

Using this function allows us to preserve class separability.

Quadratic Discriminant Analysis is based in the same assumption as LDA with class conditional densities being multivariate gaussian, but they are allowed to have different covariance matrices. This causes the decision boundary to be a quadratic function. QDA classifier involves plugging estimates for Σ_k, μ_k, and π_k into Bayes' classifier and then assigning an observation to the class for which the quantity is largest. Both LDA and QDA have excellent benefits, but the most significant aspect to consider is the bias-variance trade-off. LDA is a less flexible classifier than QDA and with a lower variance. If LDA's assumption that K classes share a common covariance matrix is off, then LDA can have an increase in bias. Thus, QDA is recommended when training sets are very large.

4.1.2 Naive Bayes

Naive Bayes is a supervised machine learning algorithm used for classification. While it's core is based in Bayes' Theorem, it's objective is to find the probability of a class (Y) given the observed features (X), written as $P(Y|X)$. Its distinguishing factor is the assumption that all predictor variables are conditionally independent from one another. Thus, naive estimation being $\prod_{i=1}^{n} P(x_i|Y)$. This means that for k=1,...,K $f_k(x) = f_{k1}(x_1) * f_{k2}(x_2) * ... * f_{kp}(x_p)$. Like most classification techniques, the data must be split into testing and training sets before the process begins. Only then can the naive Bayes assumption be made. It can then be used to obtain the expression for the posterior probability. This is a powerful algorithm because it takes into consideration the distribution of each predictor on its own as well as the joint distribution of the predictors. Naive Bayes is suggested to be used when dealing with large amounts of data. The assumption leads to some bias but also reduces variance.

4.1.3 K Nearest Neighbor

K-Nearest Neighbors (KNN) Classification is a non-parametric learning algorithm that memorizes the entire training dataset and only computes similarities during prediction (Bradly (1982)). Before starting the algorithm, it is important to split the dataset into training and test data. Once this is complete, the KNN classification algorithm can begin to be written. The first step in the KNN algorithm is to choose the number of neighbors, k. Next step is to calculate the distance between the new data point to every point in the training set. This is done by using either Euclidean Distance for representing the straight-line distance or Manhattan Distance for navigating a grid. Next you select the data points in the training dataset that have the smallest distances to the new point. Finally, it assigns the new point to the class that is most frequently among the k neighbors. Like most algorithms, KNN comes with a trade-off. A small k can cause the model to have a high variance and overfitting. On the other hand, a large k can lead to low variance and underfitting. KNN's advantages include its simplicity, versatility, and lack of time consuming training phase. One of the disadvantages most commonly noted when using KNN is that predictions can be slow because it calculates the distance to every training point.

4.1.4 Decision Tree

A decision tree is non-parametric supervised learning algorithm used for both classification and regression tasks. The primary goal of a decision tree classifier is to create a model that predicts the target variable by learning simple decision rules. A classification decision tree can be broken down into three main parts: the structure, the splitting process, and the predictions. The root node represents the start of the tree that contains the whole dataset. The internal node represents a test on an attribute. A branch represents the outcome of the test. The leaf node represents the class label. Next is the splitting process. Here, the tree is built by making choices at each step without looking ahead to see if the choice benefits the overall result. It does so by choosing the best feature and the best split point at each internal node. Its goal here is to reduce each internal node to having only one class. Finally, the algorithm classifies a new data point by starting at the root node and following the path until a leaf node is reached. This algorithm is popular for its interpret-ability and data handling. Some of its disadvantages include being prone overfitting and instability.

4.1.5 Random Forrest

A random forest is a machine learning algorithm used for both classification and regression. It is known as an ensemble method, which combines multiple models to improve performance. The main goal is to combine the predictions of multiple models to create one highly accurate model. It is also used to reduce the variance. A random forest is random because of two independent sampling steps. The first is row sampling, or R1. Row sampling creates N new subsets, or trees, using bootstrap sampling with replacement. This ensures that each tree is trained on a unique subset of the dataset. The data points that were not chosen from the original data set as part of the bootstrap training set are called Out-of-Bag samples. Typically, about a third of the original data ends up being part of the out-of-bag samples in each tree. These unused samples can be used as a validation set to estimate the model's generalization error. The second random sampling step is feature sampling, or R2. During R2, each node is split based on a handful of random features. This keeps all the trees from looking similar from the first split. Only then can each individual tree predict a class of a new data point. The class predicted by majority of trees becomes the final predicted classification.

4.2 Resampling

Resampling involves generating new samples from observed data set to estimate the performance of a model (Rizzo (2019)). Some reasons why one may choose to resample include model validation, quantifying uncertainty, handling limited data, addressing data imbalance, and model selection (Good (2005)). The process uses computational methods to gather these new samples. Because it does not use analytical formulas, resampling makes the samples highly flexible and easier to work with then the original data.

4.2.1 K-Fold Cross-Validation

Cross validation is a machine learning technique that evaluates a model's performance on unseen data by splitting the data into multiple subsets or folds (Hesterberg and Chihara (2011)). One of the most common methods of this is K-fold cross validation. This approach involves randomly dividing the set of observations into k groups, or folds, of equal size. The first fold to be created is the validation set. The validation set is set aside and used to evaluate the model's performance during each iteration. Then the mean squared error is computed on the observations in the hold out set. The hold out set is the set that is kept aside while the remaining k-1 folds are used for testing. The mean square error is computed k times. Each time, a different group of observations is used as a validation set. This process assigns a test error to each k. The k-fold cross validation estimate can be computed by using this formula.

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

An important advantage to using k-fold cross validation compared to other cross validation methods is that it often gives more accurate estimates of the test error rate. This is due to a bias-variance trade-off. Performing k-fold for k=5 or k=10 will lead to an intermediate level of bias because each training set contains only about $\frac{(k-1)n}{k}$ observations. On the other hand, the k-fold cross validation has a lower variance than other methods when it is done with k¡n. This is because the algorithm averages the outputs of fitted models that are less correlated with each other. Due to these trade-offs, most choose to perform k-fold cross validation using k=5 or k=10.

4.2.2 Bootstrapping

The bootstrap is a powerful statistical tool that is used to quantify the uncertainty associated with a given estimator or statistical learning method. For the purposes of predictive modeling, the bootstrap can be a powerful tool used to connect resampling concepts with real-world model evaluations. At its core, bootstrap samples are created by resampling with replacement from the training data set. The basic workflow starts with drawing B bootstrap samples, each with size n. The next step is to fit the model on each sample. The model performance must then be evaluated on held-out or original data. Finally, the empirical distribution of the estimates is used for inferences. Bootstrap can be used to estimate standard error, bias, and confidence intervals. As with all methods, it is important to know the bias-variance trade-off. Bootstrap is more flexible then cross-validation. However, cross validation often times has a higher variance.

5 Results

5.1 Relationships

An important relationship I wanted to highlight in my analysis was the amount of money each brand of car brought in. Using the cleaned dataset, I was able to create a plot that visualized this relationship. As seen in the plot, Jeeps most commonly had the highest totals among the other car brands. However, the highest value purchases came from Ford vehicles. The brand with the lowest total was Jetta. The rest of the distributions can be seen below.

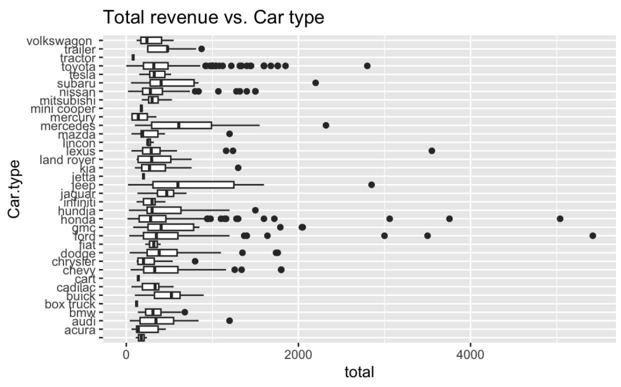


Figure 14:

Next, I wanted to look at which days of the week had the highest amount of revenue and which days of the week had the highest volume of customers. The day of the week in 2024 that had the most cars serviced was Wednesday. Monday, Wednesday, Thursday, and Saturday had between 175 and 200 customers serviced. Tuesday and Friday had between 150 and 174 customers serviced, while Sunday had between 50 to 75 customers serviced. I thought it would be interesting to compare this graph to a graph of revenue by day of the week to see if the most customers' cars serviced reflected the amount of money spent on those days. The highest amount of revenue in 2024 was brought in on Mondays. The rest of the days of the week in descending order were Thursday, Saturday, Tuesday, Sunday, Wednesday, then Friday.

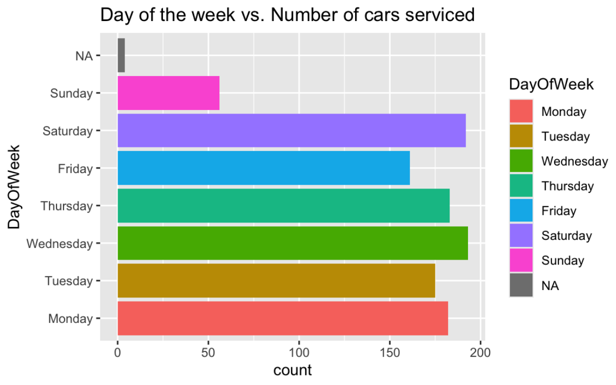


Figure 15:

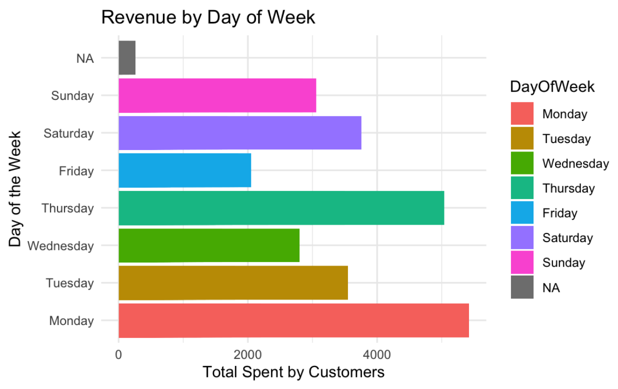


Figure 16:

Another comparison I was interested in was the relationship between the temperature of the day versus the total amount of revenue that day. I thought it would be interesting to see if inconvenient weather would cause customers to avoid going out thus decreasing the revenue or encourage them to get their cars looked at to prevent damages. The temperature in Riverside, CA is consistent, but can get very hot. This change in temperature, however, did not cause any significant changes in the revenue as seen in the plot.

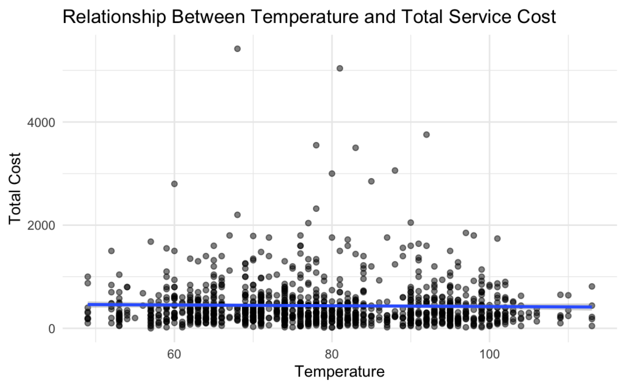


Figure 17:

Next, I was interested in seeing on average how much revenue was being brought in by each county. This was important to the company because they can then allocate a certain amount of money on advertising in the respective counties.

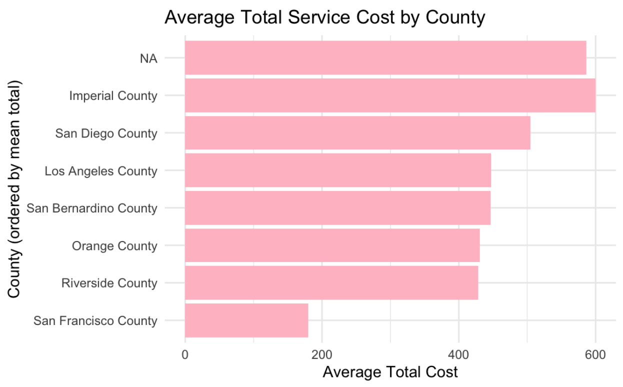


Figure 18:

Finally, I wanted to see which months brought in the most revenue. This is important for the company to know as well because they can have more employees or longer hours during their busier months. As seen in the plot, February brought in the most revenue in 2024 with well over $5,000. September was next with just shy of $5,000. The month with the least amount of revenue was July.

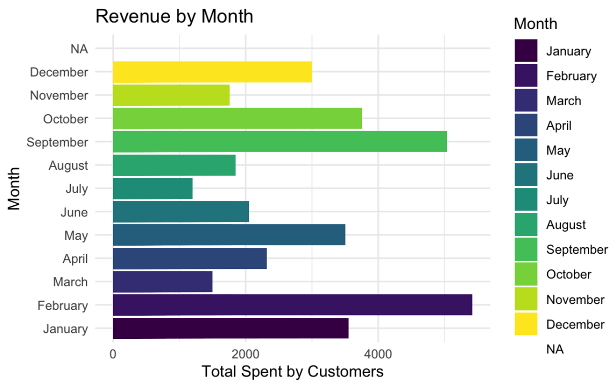


Figure 19:

5.2 LDA/QDA

After splitting my data into train and test sets, I ran an LDA model on purchased tires. The model found that 123 were correctly classified as 0, 185 were correctly classified as 1, 18 were incorrectly classified as 1 when they were actually 0, and 10 were incorrectly classified as 0 when they were actually 1. This shows that the model predicted with about 91.7 percent accuracy with 94.9 percent of buyers correctly identified and 87.2 of non-buyers correctly identified.

5.3 Naive Bayes

To begin, I used my training dataset to train the naive Bayes model. The trained data model was then used to make predictions on a new, unseen data set. The model correctly predicted that 123 customers would not buy tires, correctly predicted that 183 customers would buy tires, incorrectly predicted that 10 customers would buy tires, and incorrectly predicted that 20 customers would not buy tires. Overall, this model's classification accuracy is about 91.1 percent, meaning that the naive Bayes model correctly predicted the outcome of 91.1 percent of the customers in the test data.

5.4 K Nearest Neighbor

I decided to run a KNN classification on the tires purchased column. The model correctly predicted 131 instances where tires were not purchased, correctly predicted 183 instances where tires were purchased, incorrectly predicted 21 instances where tires were not purchased, incorrectly predicted 8 instances where tires were purchased. I then calculated the accuracy of model to be about 91.5 percent. This shows that my model has high accuracy and an appropriate balance.

5.5 Decision Tree

After splitting and training my data, I was able to run my algorithm to create a decision tree for tires purchased. The model correctly predicted 181 out of 196 actual purchases, correctly predicted 137 out of 139 actual non-purchases, incorrectly predicted 15 customers would not buy tires, and incorrectly predicted 2 customers would buy tires. The calculated accuracy was about 94.9 percent. This shows high accuracy and an efficient model.

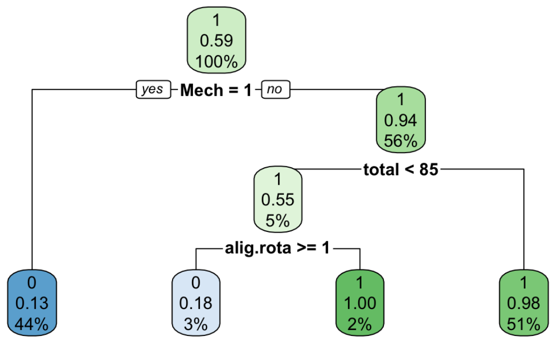
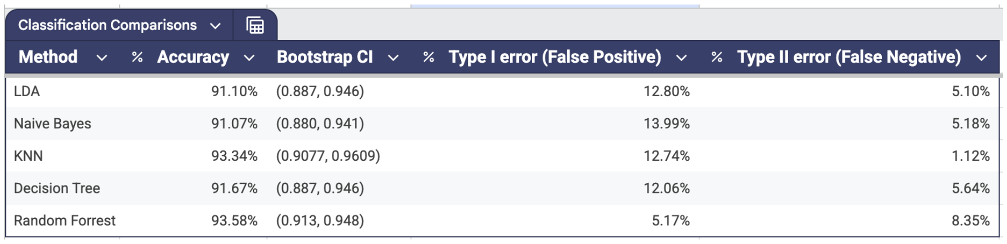


Figure 20: Classification decision tree

5.6 Random Forrest

I first split my dataset into 70 percent training data and 30 percent test data. I was then able to train my random forest model using tires purchased as the target variable and the other variables as the predictors. I set the number of decision trees in the forest to 500 and number of randomly sampled features to consider at each split to 3. I then used this model to make predictions on the unseen testing data. The model correctly predicted 181 tire purchases, correctly predicted 132 non-purchases, incorrectly predicted 15 actual purchases, and incorrectly predicted 7 purchases. I calculated the overall accuracy of my model to be about 93.4 percent.

Below is a comparison of the classifications used. On display is their accuracy, bootstrap constant interval, false positives, and false negatives. K-nearest neighbor has the highest accuracy among the methods as well as the lowest type two error. Decision tree has the second highest accuracy along with the lowest type one error.



Classification Comparisons

Method	Accuracy	Bootstrap CI	Type I error (False Positive)	Type II error (False Negative)
LDA	91.10%	(0.887, 0.946)	12.80%	5.10%
Naive Bayes	91.07%	(0.880, 0.941)	13.99%	5.18%
KNN	93.34%	(0.9077, 0.9609)	12.74%	1.12%
Decision Tree	91.67%	(0.887, 0.946)	12.06%	5.64%
Random Forrest	93.58%	(0.913, 0.948)	5.17%	8.35%

Figure 21: comparisons

5.7 K-Fold Cross-Validation

To begin, I trained a logistic regression model that included regularization, combining lasso and ridge. I proceeded to use ten-fold cross-validation. The model was trained on 9 parts and tested on the 1 left. This process was repeated ten times in order to average the performance metrics. The model was evaluated using ROC, Sens, and Spec. On average, the model had a ROC of about 0.96. The sensitivity of about 0.92 shows that it correctly identified about 92 percent of those customers who did buy tires. Specificity of 0.91 shows that it correctly identified about 91 percent of those customers who did not buy tires. The model found that the combination that produced the highest average ROC was an elastic net with 55 percent Lasso and 45 percent Ridge. The plot below visualizes where the peak performance is.

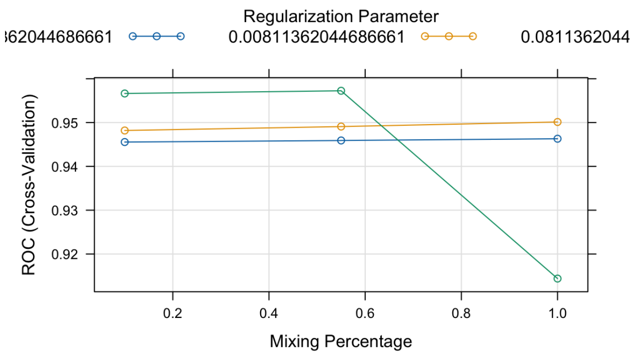


Figure 22: Regularization parameters

5.8 Bootstrapping

I started by running a bootstrapped glmnet logistic regression using a combination of Lasso and Ridge. I had the algorithm resample the data set with replacement 100 times and trained the model on each bootstrap sample. This was done to estimate the stability and variance of the model. The model shows the combination of lambda and alpha that shows the average performance across the 100 bootstrap samples. The chosen model had an alpha of 0.1 and lambda of 0.081136. This means that the best-performing combination with the highest average ROC was mainly Ridge and moderate regularization strength. The plot below visualizes where the peak performance is.

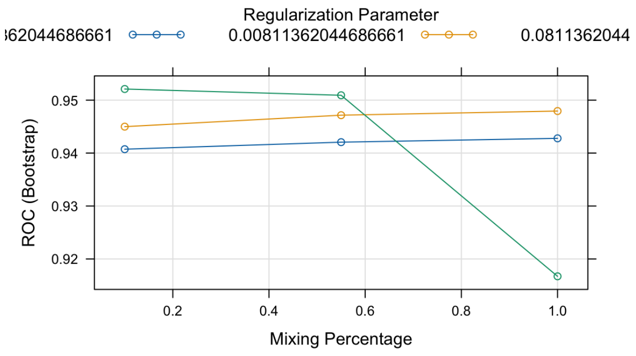


Figure 23: Regularization parameters

6 Conclusion

All American Tires provided an excellent opportunity to use the skills I have learned in this program to solve real world problems. Collecting data was by far the hardest and most time consuming part of this internship. Analyzing trends within the year's worth of invoice data was particularly interesting, as it provided direct, data-driven insights on optimal shop operations and effective advertising channels, validating management's decades of experience while allowing me to see the internal workings of the company. I found a deeper appreciation for companies like this because my answers to their questions seemed to reassure them that they were running things properly. They did not know what the data said, but their decades of experience led them to similar conclusions. Writing my methodology helped me better understand the mathematical processes behind the r functions that I use every day. Working through the predictive modeling using various classification methods, particularly comparing the performance of Random Forest against techniques like LDA/QDA, helped me hone my skills in model selection and interpretation. If I have the opportunity to work with this company in the future, I would like to compare these results to the results of different years to see how the economy affects the stability of this shop. Overall, I am grateful for this opportunity and how it allowed me to grow as a data scientist.

References

Bradly, E. (1982). *The Jackknife, the Bootstrap and Other Resampling Plans.* CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics.

Good, P. I. (2005). *Resampling Methods: A Practical Guide to Data Analysis (3rd ed.).* Birkhäuser.

Hesterberg, T. C. and Chihara, L. M. (2011). *Mathematical Statistics with Resampling and R.* Wiley & Sons, Incorporated, John.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R (2nd ed.).* Springer Texts in Statistics. Springer, New York.

Rizzo, M. L. (2019). *Statistical Computing with R (2nd ed.).* Chapman and Hall/CRC, New York.

Appendices

A R Markdown

All American Tires

Nancy Jaramillo

2025-07-02

load & clean data

library(“dplyr”) library(“timeDate”) df<-
read.csv(“~/Desktop/allamericantires/HQ.csv”,header = TRUE)

df$Date < -as.Date(df$Date, format = “%m/%d/%y”) df$DayOfWeek < -weekdays(df$Date) head(df[, c(“Date”, “DayOfWeek”)])

df$Month < -months(df$Date)

df$Car.type < -recode(df$Car.type, “hundia” = “hyundai”) df$Car.type < -tolower(df$Car.type)

df$alig.rota[is.na(df$alig.rota)]<- 0

df$Car.type < -tolower(df$Car.type)

us_holidays <- holidayNYSE(2024) df$IsHoliday < -df$Date%in%as.Date(us_holidays)

df$TiresPurchased < -ifelse(df$NT + df\$UT > 0, 1, 0)

library(“zipcodeR”) library(“dplyr”)

df$zipcode < -as.character(df$zipcode)

data(“zip_code_db”)

df_with_county <- df %>% left_join(zip_code_db, by = “zipcode”)

grouped <- df_with_county %>% group_by(county) %>% summarise(zip_codes =
list(unique(zipcode)), .groups = ‘drop’)

df_final <- df_with_county %>% select(-bounds_west, -bounds_east, -bounds_north, -
bounds_south, -median_home_value, -median_household_income, -
occupied_housing_units, -housing_units, -water_area_in_sqmi, -land_area_in_sqmi, -
population_density, -population, -area_code_list, -radius_in_miles, -timezone, -lng, -lat, -
state, -common_city_list, -post_office_city, -major_city, -zipcode_type, -predicted, -
ZipPrefix, -Tire.Size, -Tire.Brand, -Name, -zipcode, -Date, -NT, -UT)

View(df_final)

write.csv(df_final, “~/Desktop/allamericantires/df_final.csv”, row.names = FALSE)

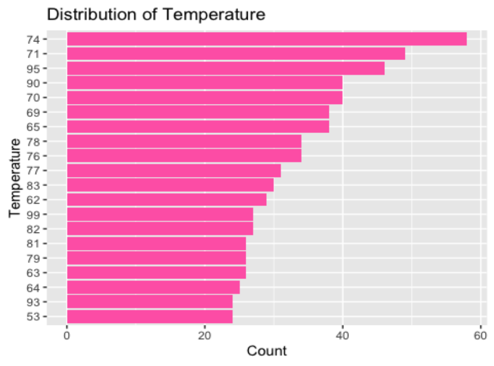
Exploratory Data Analysis

```
library(ggplot2)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.4     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(dplyr)

df_final<- read.csv("~/Desktop/math892/df_final.csv", header = TRUE)




df_final%>%
  count(Temp, sort = TRUE)%>%
  top_n(20) %>%
  ggplot(aes(x=reorder(Temp, n), y=n)) + geom_col(fill = "hotpink") +
  coord_flip()+
  labs(title = "Distribution of Temperature", x= "Temperature", y= "Count")
```

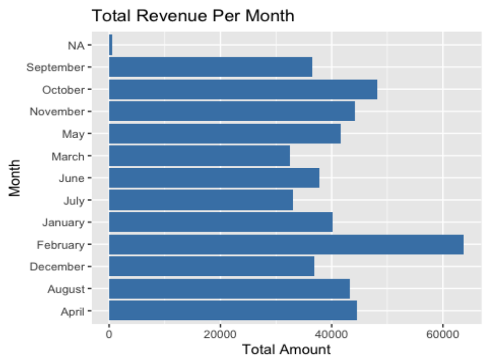
```
## Selecting by n
```



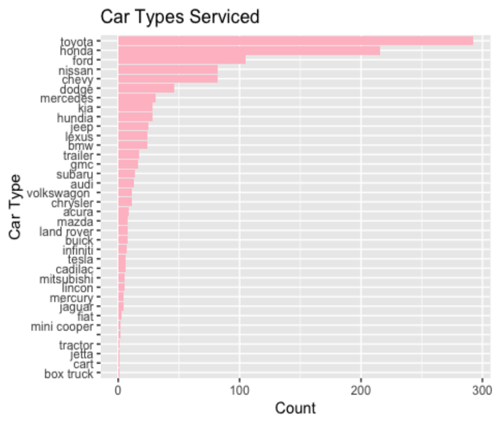
```
summary(df_final$total)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   190.0   300.0   438.8   520.0  5420.0
```

```r
df_final$total <- as.numeric(df_final$total)

df_summary <- df_final %>%
  group_by(Month) %>%
  summarise(total = sum(total, na.rm = TRUE))

ggplot(df_summary, aes(x = Month, y = total)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Total Revenue Per Month",
       x = "Month", y = "Total Amount")
```

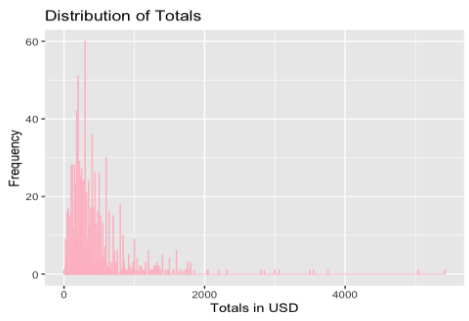


```r
df_final %>%
  count(Car.type, sort=TRUE) %>%
  #top_n(10)%>%
  ggplot(aes(x=reorder(Car.type, n), y = n)) + geom_col(fill = "pink") +
coord_flip() +
  labs(title = "Car Types Serviced", x = "Car Type", y = "Count")
```

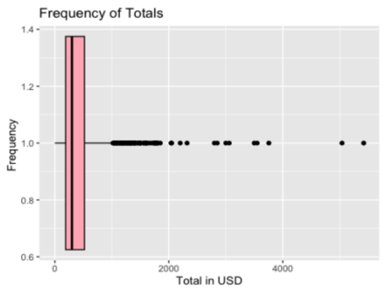


```r
summary(df_final$total)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   190.0   300.0   438.8   520.0  5420.0
```

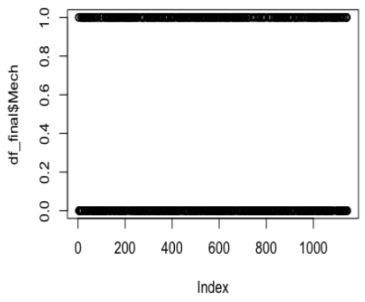
```r
ggplot(df_final, aes(x = total)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "pink") +
  labs(title = "Distribution of Totals", x = "Totals in USD", y =
"Frequency")
```



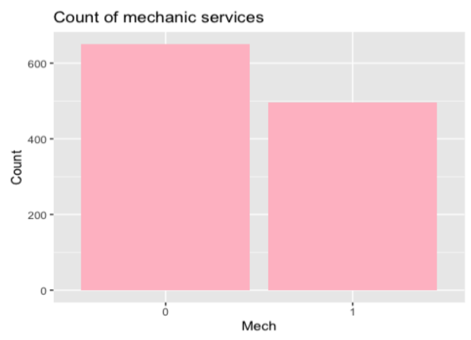
```r
ggplot(df_final, aes(x = total, y = frequency(total))) +
  geom_boxplot(fill = "lightpink", color = "black") +
  labs(title = "Frequency of Totals",
       x = "Total in USD",
       y = "Frequency ")
```



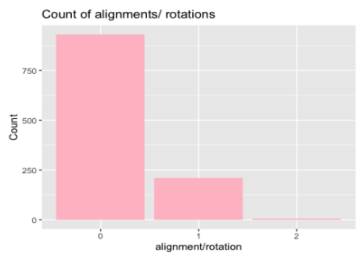
```r
plot(df_final$Mech)
```



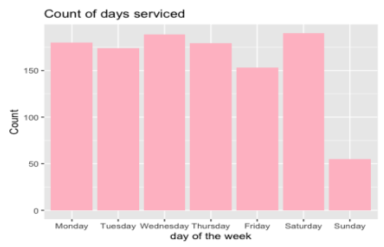
```r
ggplot(df_final, aes(x = factor(Mech))) +
  geom_bar(fill = "pink") +
  labs(title = "Count of mechanic services",
       x = "Mech", y = "Count")
```



```
ggplot(df_final, aes(x = factor(alig.rota))) +
  geom_bar(fill = "pink") +
  labs(title = "Count of alignments/ rotations",
       x = "alignment/rotation", y = "Count")
```

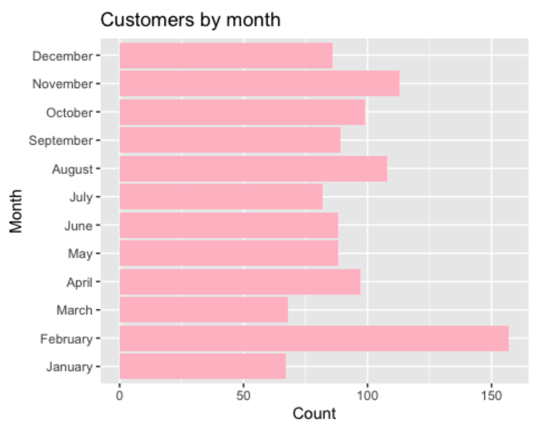


```
ggplot(na.omit(df_final), aes(x = factor(DayOfWeek,
                                levels = c("Monday", "Tuesday", "Wednesday",
                                           "Thursday", "Friday", "Saturday",
"Sunday")))) +
  geom_bar(fill = "pink") +
  labs(title = "Count of days serviced",
       x = "day of the week", y = "Count")
```

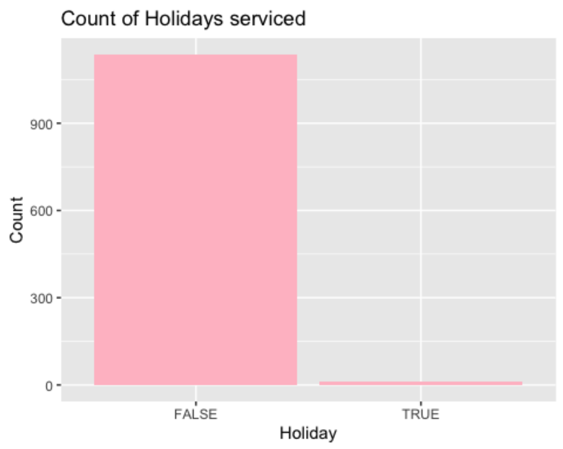


```
ggplot(df_final[!is.na(df_final$Month), ], aes(x = factor(Month,
                                levels = c("January", "February", "March",
"April",
                                           "May", "June", "July", "August",
                                           "September", "October",
"November", "December")))) +
```

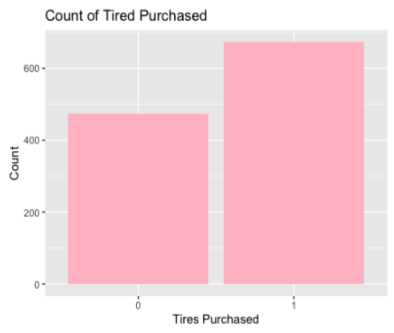
```r
  geom_bar(fill = "pink") +
  coord_flip() +
  labs(title = "Customers by month",
       x = "Month", y = "Count")
```



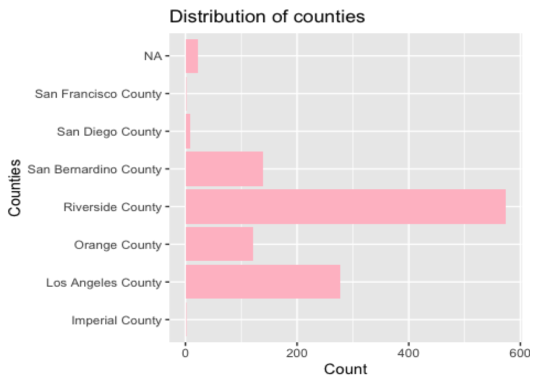
```r
ggplot(df_final, aes(x = factor(IsHoliday))) +
  geom_bar(fill = "pink") +
  labs(title = "Count of Holidays serviced",
       x = "Holiday", y = "Count")
```



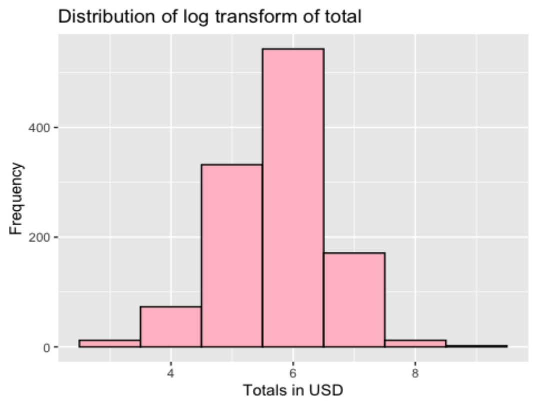
```r
ggplot(df_final, aes(x = factor(TiresPurchased))) +
  geom_bar(fill = "pink") +
  labs(title = "Count of Tired Purchased",
       x = "Tires Purchased", y = "Count")
```



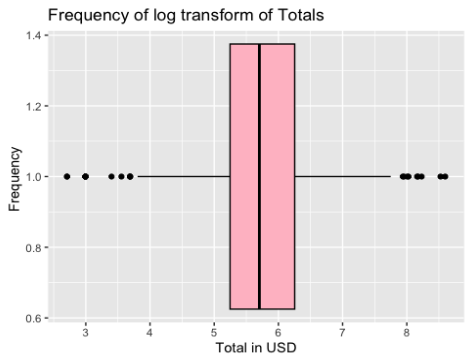
```
ggplot(df_final, aes(x = factor(county))) +
  geom_bar(fill = "pink") +
  coord_flip()+
  labs(title = "Distribution of counties",
       x = "Counties", y = "Count")
```



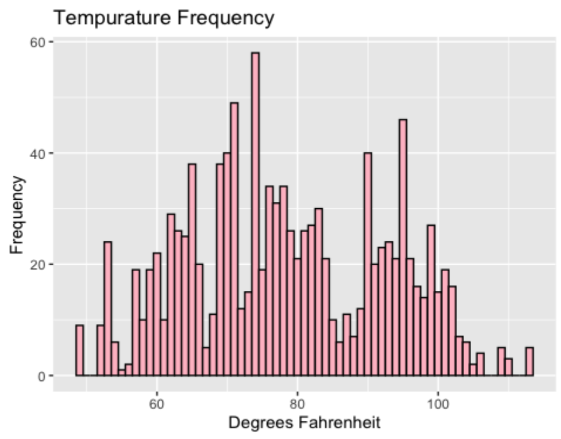
```
log_total<- log(df_final$total)
summary(log_total)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -Inf   5.247   5.704    -Inf   6.254   8.598

ggplot(df_final, aes(x = log_total)) +
  geom_histogram(binwidth = 1, fill = "pink", color = "black") +
  labs(title = "Distribution of log transform of total", x = "Totals in USD",
y = "Frequency")

## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_bin()`).
```



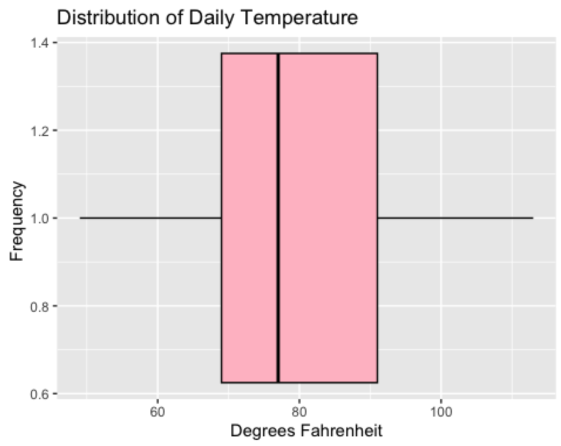
```
ggplot(df_final, aes(x = log_total, y = frequency(log_total))) +
  geom_boxplot(fill = "pink", color = "black") +
  labs(title = "Frequency of log transform of Totals",
       x = "Total in USD",
       y = "Frequency ")

## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_boxplot()`).
```



```
ggplot(df_final, aes(x = Temp)) +
  geom_histogram(binwidth = 1, fill = "pink", color = "black") +
  labs(title = "Tempurature Frequency", x = "Degrees Fahrenheit", y =
"Frequency")
```



```r
ggplot(df_final, aes(x = Temp, y = frequency(Temp))) +
  geom_boxplot(fill = "pink", color = "black") +
  labs(title = "Distribution of Daily Temperature",
       x = "Degrees Fahrenheit",
       y = "Frequency ")
```



visualization

library(ggplot2) library(dplyr)

df_final %>% group_by(Date) %>% summarise(total_services = sum(total)) %>% ggplot(aes(x=Date, y=total_services)) + geom_line(col="steelblue") + labs(title = "Total Revenue Amount over Time", x = "Date", y = "Total Amount")

df %>% count(Car.type, sort=TRUE) %>% top_n(10)%>% ggplot(aes(x=reorder(Car.type, n), y = n)) + geom_col(fill = "pink") + coord_flip() + labs(title = "Top 10 Car Types Serviced", x = "Car Type", y = "Count") df %>% filter(Tire.Brand != “Unknown”) %>% count(Tire.Brand, sort=TRUE) %>% top_n(10) %>% ggplot(aes(x= reorder(Tire.Brand, n), y=n)) + geom_col(fill = “purple”) + coord_flip() + labs(title = “Top 10 Tire Brands Sold”, x = “Tire Brand”, y = “Count”)

df %>% filter(Tire.Size != “Unknown”) %>% count(Tire.Size, sort = TRUE) %>% top_n(10) %>% ggplot(aes(x=reorder(Tire.Size, n), y=n)) + geom_col(fill = “hotpink”) + coord_flip() + labs(title = “Top 10 Tire Sizes Sold”, x = “Tire Size”, y = “Count”)

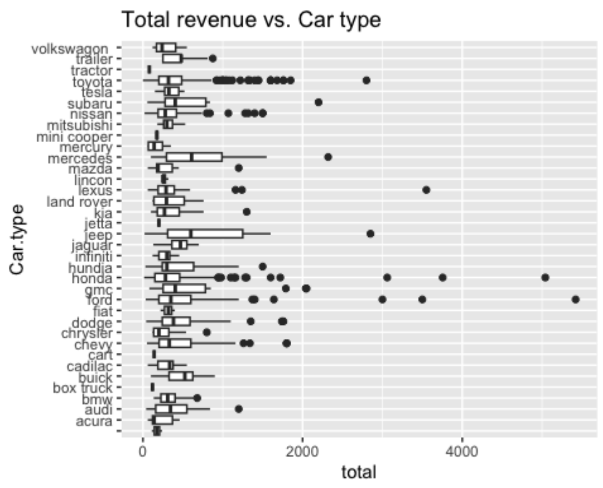
df$ZipPrefix < -substr(as.character(df$Zip), 1, 3)

ggplot(df, aes(x = factor(ZipPrefix))) + geom_bar(fill = “red”) + labs(title = “ZIP Code Prefixes (first 3 digits)”, x = “ZIP Prefixes”, y = “Count”) + theme_minimal()

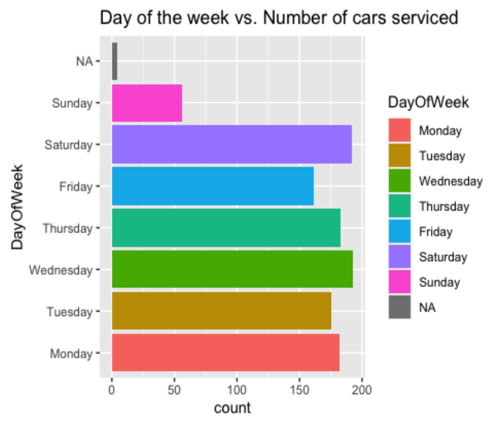
ggplot(df_final, aes(x = county)) + geom_bar(fill = “steelblue”) + coord_flip() + labs(title = “Counties by Count”, x = “County”, y = “Count”) + theme_minimal()

relationship visualizations

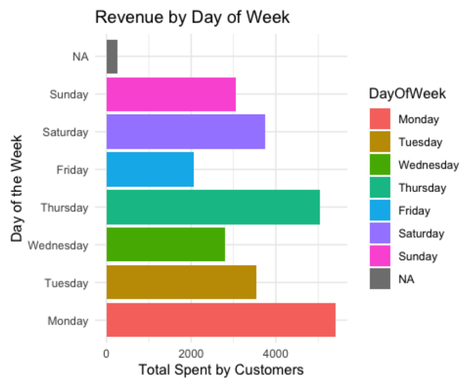
```
library(ggplot2)
library(dplyr)

df <- df_final %>%
  mutate(
    DayOfWeek = factor(DayOfWeek,
                       levels =
c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday")),
    Month = factor(Month, levels = month.name)
  )

ggplot(df, aes(x = Car.type, y = total)) + geom_boxplot() +coord_flip() +
labs(title = "Total revenue vs. Car type")
```



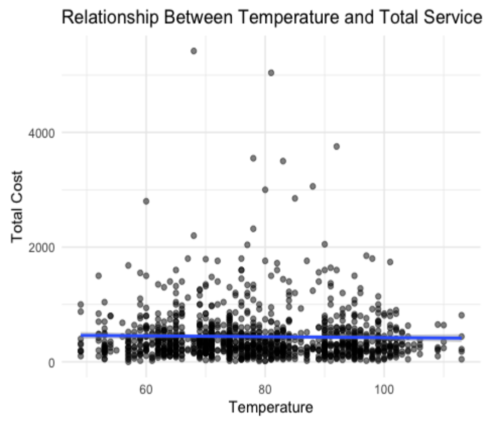
```r
ggplot(df, aes(x = DayOfWeek, fill = DayOfWeek)) + coord_flip() +geom_bar() +
labs(title = "Day of the week vs. Number of cars serviced")
```



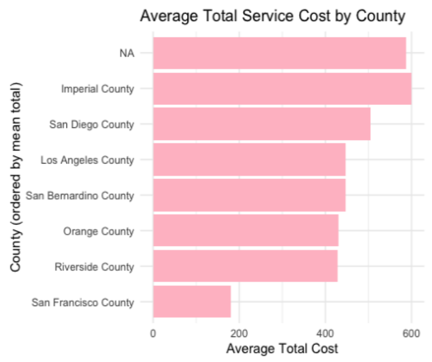
```r
ggplot(df, aes(x = DayOfWeek, y = total, fill = DayOfWeek)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  labs(title = "Revenue by Day of Week", x = "Day of the Week", y = "Total
Spent by Customers") +
  theme_minimal()
```



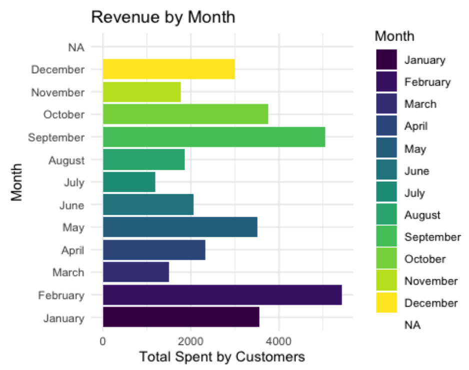
```r
ggplot(df, aes(x = Temp, y = total)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = TRUE) +
  labs(title = "Relationship Between Temperature and Total Service Cost",
       x = "Temperature",
       y = "Total Cost") +
  theme_minimal()

## `geom_smooth()` using formula = 'y ~ x'
```



```r
ggplot(df, aes(x = reorder(county, total, FUN = mean), y = total)) +
  stat_summary(fun = mean, geom = "bar", fill = "pink") +
  coord_flip() +
  labs(title = "Average Total Service Cost by County",
       x = "County (ordered by mean total)",
       y = "Average Total Cost") +
  theme_minimal()
```



```r
df_final$Month <- factor(df_final$Month,
                         levels = month.name,  # January to December
                         ordered = TRUE)

ggplot(df_final, aes(x = Month, y = total, fill = Month)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  labs(title = "Revenue by Month", x = "Month", y = "Total Spent by
Customers") +
  theme_minimal()
```



LDA/QDA

```
library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

library(tidyverse)
library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(mvtnorm)
library(ggplot2)
df_final$TiresPurchased <- as.factor(df_final$TiresPurchased)
df_final$Temp <- as.factor(df_final$Temp)
df_final$DayOfWeek <- as.factor(df_final$DayOfWeek)
df_final$Month <- as.factor(df_final$Month)
df_final$county <- as.factor(df_final$county)
df_final$IsHoliday <- as.numeric(df_final$IsHoliday)
df_final<- na.omit(df_final)

set.seed(11)
train_ind <- sample(seq_len(nrow(df_final)), size = 0.7 * nrow(df_final))
train_data <- df_final[train_ind, ]
test_data <- df_final[-train_ind, ]
```

```r
num_features <- names(train_data)[sapply(train_data, is.numeric)]

train_means <- sapply(train_data[num_features], mean)
train_sds <- sapply(train_data[num_features], sd)

train_data[num_features] <- scale(train_data[num_features], center =
train_means, scale = train_sds)

test_data[num_features] <- scale(test_data[num_features], center =
train_means, scale = train_sds)

lda_mod <- lda(TiresPurchased~., data = df_final)
lda_pred <- predict(lda_mod, newdata = test_data)
table(Predicted = lda_pred$class, Actual = test_data$TiresPurchased)

##          Actual
## Predicted   0   1
##         0 123  18
##         1  10 185

table(train_data$TiresPurchased)

##
##   0   1
## 331 453
```

naive bayes

```r
library(e1071)

nbayes_mod <- naiveBayes(TiresPurchased~., data = train_data)
nbayes_pred <- predict(nbayes_mod, newdata = test_data)
table(Predicted = nbayes_pred, Actual = test_data$TiresPurchased)

##          Actual
## Predicted   0   1
##         0 123  20
##         1  10 183

mean(nbayes_pred == test_data$TiresPurchased)

## [1] 0.9107143
```

knn

```r
library(class)
library(caret)

df_knn <- df_final[, !names(df_final) %in% c("Car.type", "DayOfWeek",
"Month", "county")]
df_knn <- na.omit(df_knn)
```

```r
set.seed(111)
splitInd <- createDataPartition(df_knn$TiresPurchased, p = 0.7, list = FALSE)
train_knn <- df_knn[splitInd, ]
test_knn <- df_knn[-splitInd, ]

num_cols <- sapply(train_knn, is.numeric)
traink_scaled <- as.data.frame(scale(train_knn[, num_cols]))
testk_scaled <- as.data.frame(scale(test_knn[, num_cols], center =
attr(scale(train_knn[, num_cols]), "scaled:center"), scale =
attr(scale(train_knn[, num_cols]), "scaled:scale")))

traink_scaled$TiresPurchased <- train_knn$TiresPurchased
testk_scaled$TiresPurchased <- test_knn$TiresPurchased

train_x <- traink_scaled[, -which(names(traink_scaled) == "TiresPurchased")]
test_x <- testk_scaled[, -which(names(testk_scaled) == "TiresPurchased")]
train_y <- traink_scaled$TiresPurchased
test_y <- testk_scaled$TiresPurchased

knn_pred <- knn(train = train_x, test = test_x, cl = train_y, k = 5)
table(Predicted = knn_pred, Actual = test_y)
```

```
##          Actual
## Predicted   0   1
##         0 137  20
##         1   2 176
```

```r
mean(knn_pred == test_y)
```

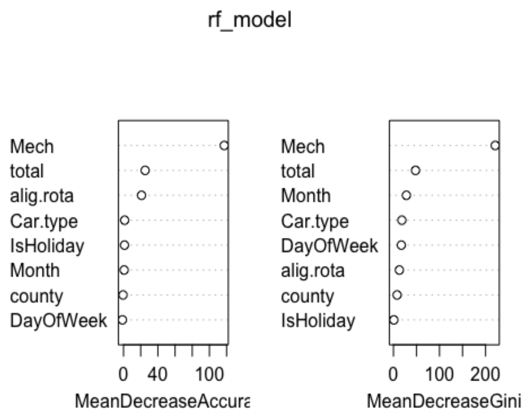
```
## [1] 0.9343284
```

random forest

```r
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(caret)

df_rf <- na.omit(df)
df_rf <- df_rf[, !names(df_rf) %in% c("Temp")]

set.seed(222)
split <- createDataPartition(df_rf$TiresPurchased, p = 0.7, list = FALSE)
train_rf <- df_rf[split, ]
test_rf <- df_rf[-split, ]
X_train=train_rf[,c(2,3,4,5,9)]
rf_model <- randomForest(X_train, factor(train_rf$TiresPurchased),  ntree =
500,  importance = TRUE, type="classification")
rf_model <- randomForest(factor(train_rf$TiresPurchased)~., data=train_rf,
ntree = 500, importance = TRUE, type="classification")

rf_pred <- predict(rf_model, newdata = test_rf)
#table(predicted = rf_pred, actual = test_rf$TiresPurchased)
mean(rf_pred == test_rf$TiresPurchased)

## [1] 0.9017857

importance(rf_model)

##                      0            1 MeanDecreaseAccuracy MeanDecreaseGini
## Car.type    -0.1346049   2.32793231            1.3217184       18.2760772
## Mech       112.6552232 104.59052093          117.2458250      221.0277920
## alig.rota   21.6169827   7.72944418           20.9680018       12.8985490
## total       22.3850157  13.09139853           25.1361716       47.9500730
## DayOfWeek   -1.7618911   0.35047175           -1.2561195       17.0936852
## Month       -0.4882829   1.81503490            0.7095097       28.0830253
## IsHoliday    1.3564349   0.27576774            1.1285744        0.8917266
## county      -0.8534657   0.09195351           -0.5840301        7.9480230

varImpPlot(rf_model)
```

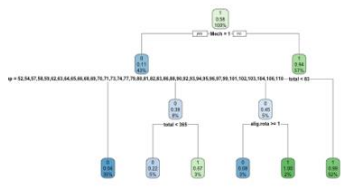


```r
rf_model$confusion
```

```
##     0   1 class.error
## 0 312  17  0.05167173
## 1  38 417  0.08351648
```

decision trees

```r
library(rpart)
library(rpart.plot)

df_tree<- na.omit(df_final)

set.seed(22)
train_tree <- df_tree[split, ]
test_tree <- df_tree[-split, ]

tree_model <- rpart(TiresPurchased~., data = train_tree, method = "class")
rpart.plot(tree_model, extra = 106)
```



```r
test_tree$DayOfWeek <- factor(test_tree$DayOfWeek, levels =
levels(train_tree$DayOfWeek))
test_tree$Month <- factor(test_tree$Month, levels = levels(train_tree$Month))
test_tree$county <- factor(test_tree$county, levels =
levels(train_tree$county))
test_tree$Car.type <- factor(test_tree$Car.type, levels =
levels(train_tree$Car.type))

tree_pred <- predict(tree_model, newdata = test_tree, type = "class")
table(predicted = tree_pred, actual = test_tree$TiresPurchased)
```

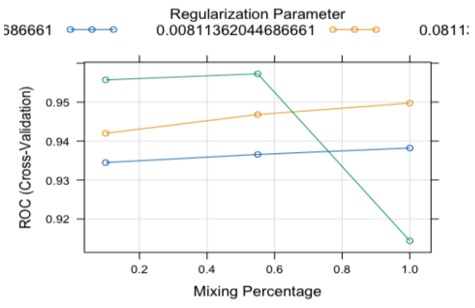
```
##          actual
## predicted   0   1
##         0 124  17
##         1  11 184
```

```r
mean(tree_pred == test_tree$TiresPurchased)
```

```
## [1] 0.9166667
```

resampling: k-fold cross-validation

```
library(caret)

df_final <- na.omit(df_final)

set.seed(2)

cv_cont <- trainControl(method = "cv", number = 10, classProbs = TRUE,
summaryFunction = twoClassSummary, savePredictions = "final")

df_final$TiresPurchased <- factor(df_final$TiresPurchased, labels = c("No",
"Yes"))

glm_cv <- train(TiresPurchased~., data = df_final, method = "glmnet", family
= "binomial", trControl = cv_cont, metric = "ROC")
print(glm_cv)

## glmnet
##
## 1120 samples
##    9 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1007, 1008, 1009, 1008, 1008, 1007, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda       ROC        Sens       Spec
##   0.10   0.000811362  0.9344786  0.8684089  0.9070396
##   0.10   0.008113620  0.9419839  0.8835338  0.9085781
##   0.10   0.081136204  0.9557083  0.9051341  0.9101166
##   0.55   0.000811362  0.9365599  0.8705828  0.9100932
##   0.55   0.008113620  0.9468135  0.8965310  0.9085781
##   0.55   0.081136204  0.9572549  0.9202128  0.9085781
##   1.00   0.000811362  0.9382114  0.8705828  0.9116084
##   1.00   0.008113620  0.9497271  0.9029602  0.9116317
##   1.00   0.081136204  0.9143954  0.9202128  0.9085781
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.55 and lambda =
0.0811362.

plot(glm_cv)
```

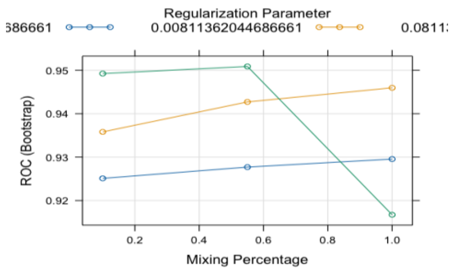


bootstrapping

```r
library(caret)

set.seed(100)

boot_control <- trainControl(
  method = "boot",
  number = 100,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)

boot_glm <- train(
  TiresPurchased ~ .,
  data = df_final,
  method = "glmnet",
  family = "binomial",
  trControl = boot_control,
  metric = "ROC"
)

print(boot_glm)
```

```
## glmnet
##
## 1120 samples
##    9 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 1120, 1120, 1120, 1120, 1120, 1120, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda       ROC        Sens       Spec
##   0.10   0.000811362  0.9251041  0.8397942  0.8995629
##   0.10   0.008113620  0.9358085  0.8672732  0.9093289
##   0.10   0.081136204  0.9492401  0.9029038  0.9127133
##   0.55   0.000811362  0.9276975  0.8475274  0.9028264
##   0.55   0.008113620  0.9427057  0.8862186  0.9136810
##   0.55   0.081136204  0.9508755  0.9177740  0.9101361
##   1.00   0.000811362  0.9295476  0.8517495  0.9051764
##   1.00   0.008113620  0.9459634  0.8941513  0.9141900
##   1.00   0.081136204  0.9167210  0.9198789  0.9100552
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.55 and lambda =
0.0811362.
```

```r
plot(boot_glm)
```


Regularization Parameter
ROC (Bootstrap)
Mixing Percentage